

Podemos criar **Threads** em Aplicações **Java** rodando em Servidores **Web**?

Se sim, como?

Se não, por quê?



Você pode compartilhar este trabalho.

Apenas mantenha os créditos e seja explícito quanto a eles.
~~Parte chata~~ Detalhes ao lado.

Esta apresentação de Rinaldo Pitzer Júnior está licenciada com uma Licença Creative Commons - Atribuição 4.0 Internacional.

Canal do Autor: <https://youtube.com/rinaldodev>

Página do Autor: <https://rinaldo.dev/>

Baseado no trabalho disponível em

<https://docs.google.com/presentation/d/1-GL8frvtM5qPPSSJGcTx8Nhi3DGDg5eB-m-RiILNVbw/edit?usp=sharing>.

Detalhes da licença disponíveis em:

https://creativecommons.org/licenses/by/4.0/deed.pt_BR



Conteúdo relacionado.

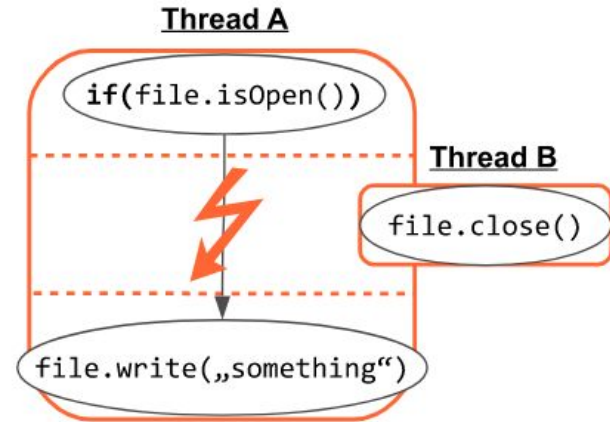
[Playlist de Multithread](#)

Futura playlist de multithread em servidores web.

**Primeiro, vamos falar de
concorrência.**

Relembrando...

Acesso a um recurso compartilhado.



**Use qualquer ferramenta do
Java para lidar com
concorrência.**



Quais?

As que falei na minha [Playlist!](#) :P

Synchronized

Coleções thread-safe

Classes Atômicas

Volatile

Lock

CyclicBarrier


CountDownLatch

Semaphore

...

Agora sim, threads.

**Em servidores web, suas
threads são gerenciadas.**



O que é gerenciado?

Muita coisa.

Injeção de dependência

@Inject

@EJB

@ResourceContext

Contexto transacional

@Transacional

@TranctionManagement

Contexto de segurança

@Roles

@PermitAll

Processamento de outras anotações

Coisas que a gente nem sabe



JSR 236: Concurrency Utilities for Java EE

“java.util.Timer, java.lang.Thread and the thread pool creation APIs from the Java SE concurrency utilities (JSR-166) in the java.util.concurrent package should never be used within managed environments, as it creates threads outside the purview of the container.”



JSR 236: Utilitários de Concorrência para Java EE

“As classes `java.util.Timer`, `java.lang.Thread` e as APIs para criação de pool de threads dos utilitários de concorrência do Java SE (JSR-166) no pacote `java.util.concurrent` nunca devem ser usadas em ambientes gerenciados, pois criam threads desconhecidas pelo container.”

Se você criar threads da forma usual, elas *não* serão gerenciadas.



E daí?

Você perde tudo isso aí.

Injeção de dependência

@Inject

@EJB

@ResourceContext

Contexto transacional

@Transacional

@TranctionManagement

Contexto de segurança

@Roles

@PermitAll

Processamento de outras anotações

Coisas que a gente nem sabe

Tá, entendi, mas eu quero minhas threads.



Vamos por partes.

O que você usa?

Uso Java/Jakarta EE.

WildFly/JBoss, TomEE, Payara, GlassFish,
...

Uso um container enxuto.

Tomcat, Jetty, ...

Sou descolado.

Quarkus, Spring, ...

Só quero rodar um jarzim (.jar)

Java

**Sou comportado, uso
Java/Jakarta EE.**

Muita coisa é gerenciada.



Java/Jakarta EE: suas principais opções.

ManagedExecutorService

ManagedScheduledExecutorService

@Schedule/@Timeout de EJB


@Asynchronous de EJB

MDBs (Message-Driven Beans)



ManagedExecutorService

Versão gerenciada do ExecutorService.



ManagedScheduledExecutorService

Versão gerenciada do ScheduledExecutorService.



@Schedule/@Timeout de EJB

Agendamento de tarefas com intervalo fixo ou ritmo fixo.



@Asynchronous de EJB

Chamadas assíncronas para métodos.



MDBs (Message-Driven Beans)

Execução após receber uma mensagem.



Como escolho?

Mais liberdade, mais responsabilidade.

ManagedExecutorService

ManagedScheduledExecutorService

Menos responsabilidade, menos liberdade.

@Schedule/@Timeout de EJB

@Asynchronous de EJB

MDBs (Message-Driven Beans)

**Sou independente, uso um
container enxuto.**

Pouca coisa é gerenciada.



Container enxuto: suas principais opções.

Quartz

ScheduledExecutorService

Ferramentas do Java SE*



Quartz

Framework de execução de tarefas.



ScheduledExecutorService

Crie no ServletContextListener ao iniciar o servidor.



Ferramentas do Java SE*

*~~Não use~~ Use com cuidado.

Sou descolado.

~~Se vira.~~ Leia a documentação.



Descolados, consultem seus frameworks.

Quarkus

Sem cluster?

<https://quarkus.io/guides/scheduler>

Com cluster?

<https://quarkus.io/guides/quartz>

Spring

<https://spring.io/guides/gs/scheduling-tasks/>

<https://spring.io/guides/gs/batch-processing/>

<https://www.baeldung.com/spring-scheduled-tasks>

<https://www.baeldung.com/introduction-to-spring-batch>

Tá curtindo?



Considere tornar-se um membro do canal e ter benefícios exclusivos. Detalhes abaixo.



**Sou um humilde consciente, só
quero executar meu jar.**

Parabéns.





Rodar um jar: use um agendador externo.

Ferramentas de agendamento

Agendamento do SO

Isso é tudo?

Não, espera que tem mais.

Posso usar Streams Paralelos?

~~Não use.~~ Depende.



Streams Paralelos

Feitos para uso intensivo de Processador, não de I/O.

Usam uma thread pool única.

Threads não gerenciadas.

Discussão: <http://mail.openjdk.java.net/pipermail/lambda-dev/2013-April/009334.html>

Posso usar CompletableFuture?

Se fizer certo, pode ser que dê certo.



CompletableFuture

Threads não gerenciadas por padrão.

Use junto do `ManagedExecutorService`.

```
CompletableFuture.supplyAsync(() -> { //... }, managedExecutorService)
```

Considere `@Async` do EJB.

— Agora acabou, né?

Não, ainda tem mais, acredita?



O que não falei.

JSR 352: Batch Applications for the Java Platform.

JSR 236: ManagedThreadFactory e ContextService.

...

Curtiu?

Considere tornar-se um membro do canal e ter benefícios exclusivos. Detalhes abaixo.





Você pode compartilhar este trabalho.

Apenas mantenha os créditos e seja explícito quanto a eles.
~~Parte chata~~ Detalhes ao lado.

Esta apresentação de Rinaldo Pitzer Júnior está licenciada com uma Licença Creative Commons - Atribuição 4.0 Internacional.

Canal do Autor: <https://youtube.com/rinaldodev>

Página do Autor: <https://rinaldo.dev/>

Baseado no trabalho disponível em


<https://docs.google.com/presentation/d/1-GL8frvtM5qPPSSJGcTx8Nhi3DGDg5eB-m-RiILNVbw/edit?usp=sharing>.

Detalhes da licença disponíveis em:

https://creativecommons.org/licenses/by/4.0/deed.pt_BR

Agora acabou.





Inscreva-se!

Comente!

Compartilhe!

